Multirate Signal Processing

November 22, 2018

Christian Knoll, christian.knoll@tugraz.at, Josef Kulmer, kulmer@tugraz.at
Franz Pernkopf, pernkopf@tugraz.at
Markus Rauter, Christian Feldbauer, Klaus Witrisal, Erhard Rank
Signal Processing and Speech Communication Laboratory, www.spsc.tugraz.at
Graz University of Technology

Abstract

In single-rate systems, only one sampling rate is used throughout a digital signal processing system, whereas in multirate systems the sampling rate is changed at least once. This laboratory deals with the realization and analysis of multirate systems. In the first experiment we will examine the effects of decimation and interpolation. Furthermore, a small multirate signal processing system is built and the quality of the reconstructed output signal will be discussed.

1 Decimation and Interpolation

Decimation and interpolation are processes that transform a discrete-time signal with sampling rate f_s to another discrete-time signal with a new sampling rate f_s'. A decimator realizes a sampling rate conversion (down-sampling) by an integer factor M: $f_s' = f_s/M$, by using only every M^{th} input signal sample in the output signal. The output signal is at a lower sampling rate, and thus has a lower bandwidth ($f_s'/2$) than the input signal ($f_s/2$). To avoid aliasing and thus ensure correct reproduction of the signal spectrum in $f_s'/2$, the input signal has to be low-pass filtered before sampling rate conversion. The block diagram of a decimator and example spectra of input, intermediate, and output signal for a decimation factor $M = 3$ are shown in Figure 1. Note the different scaling of the normalized frequency θ' axis in the output signal spectrum.

For interpolation a sampling rate conversion (up-sampling) by an integer factor L is achieved by inserting $L - 1$ zero samples between adjacent input signal samples. The resulting signal has a sampling rate $f_s' = Lf_s$, and a spectrum that includes periodic images of the original signals spectrum. To correctly reproduce the spectrum of the input signal (in the frequency range up to $f_s/2$) subsequent low-pass filtering is necessary to avoid imaging [1]. The block diagram of an interpolator, example signals and spectra for an interpolation factor $L = 3$ are shown in Figure 2.

Fractional changes $\frac{L}{M}$ of the sampling rate can be achieved by combining a decimator with factor M with an interpolator with factor L. For block processing the realization of decimation/interpolation is trivial, as long as the block size is an integer multiple of the decimation/interpolation factor. Otherwise, further considerations are necessary (scheduling, block redistribution, delay, etc.).

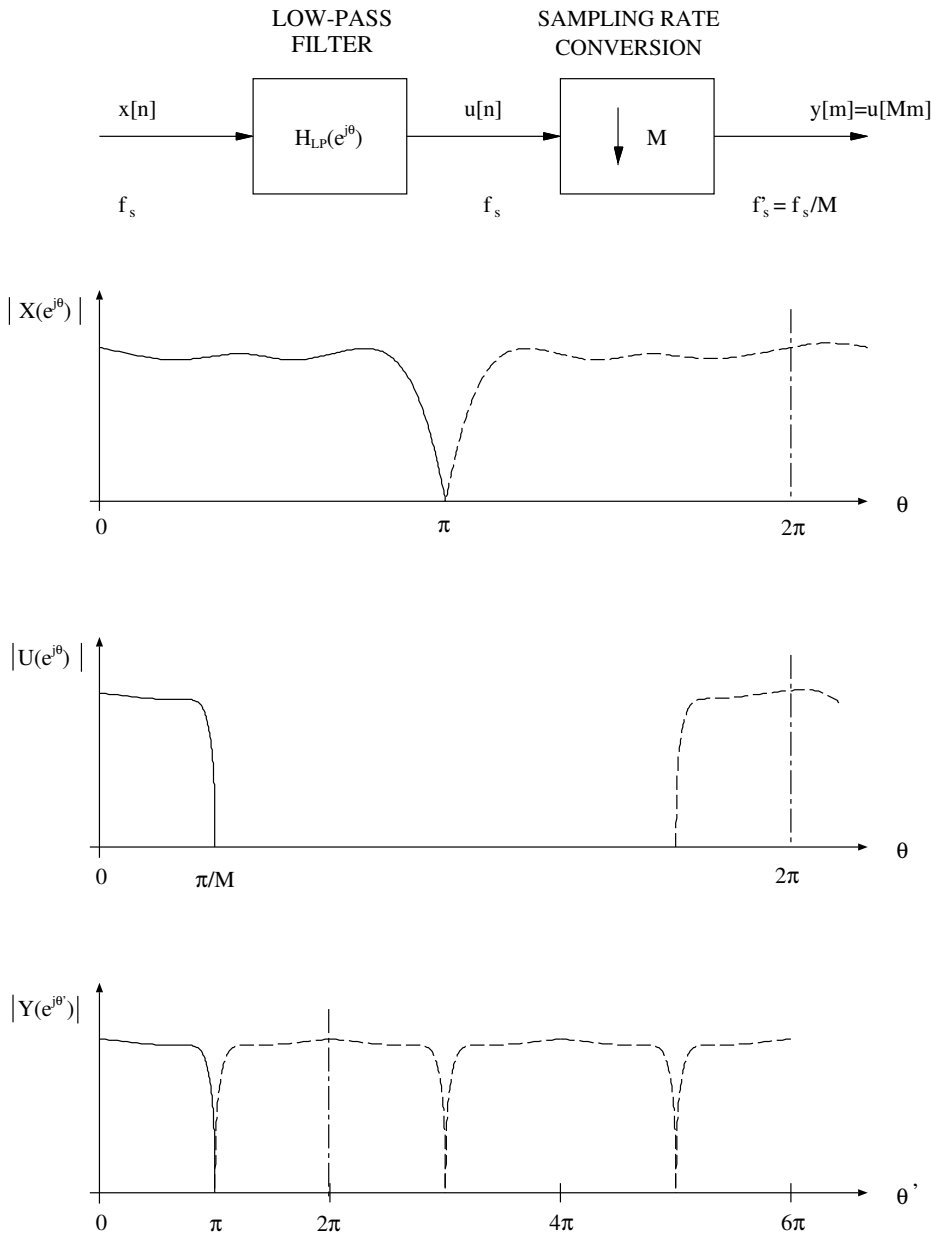

Figure 1: Block diagram of a decimator and example signal spectra for M=3.

1.1 Experiment 1: Analyze the Effects of Decimation and Interpolation

Equipment: PC, raspberry pi, signal generator, oscilloscope
Software: NetBeans, MATLAB, download and unzip `Unit6.zip` from www.spsc.tugraz.at

1. Load the project file in NetBeans.

2. Look at the file `main.cpp` . A chain of a decimator and an interpolator (without filters) is already defined there. The sampling rate conversion should be set to the factor of 4, defined in the constants `DECFACT` and `INTFACT` . Add the required parameters to

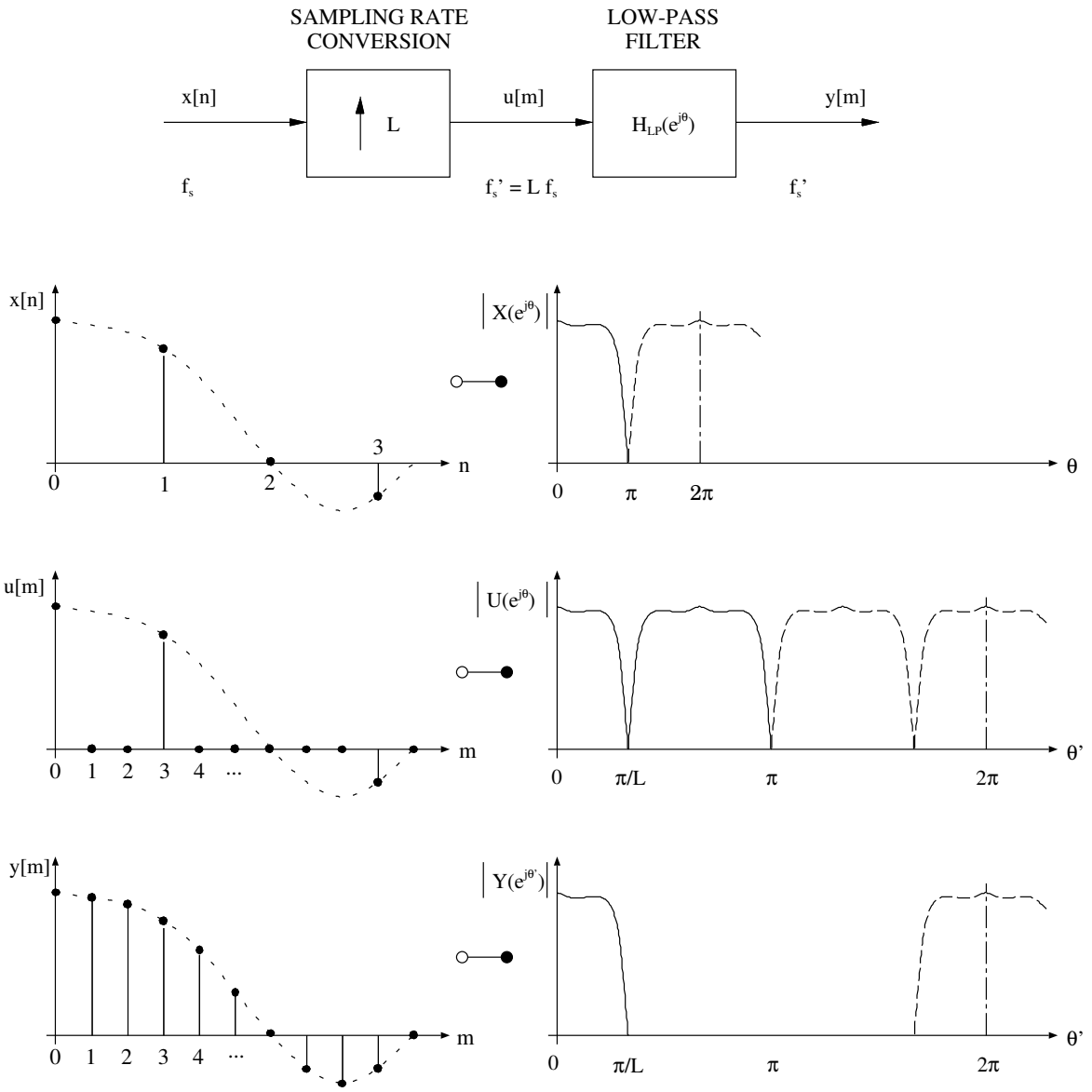

Figure 2: Block diagram of an interpolator, example signals and spectra for L=3.

`block.decimate` and `block.interpolate` . Do not add the filters at this step. Build and run the program.

3. Apply a sine-wave signal (at, e.g., 700 Hz, $V_{PP} = 0.5\,V$) to the system and observe the output spectrum on the oscilloscope. Vary the frequency of the sine-wave signal. Explain your observations and add figures to your report.

4. Introduce a low-pass interpolation filter at the output (see Figure 2). Choose a FIR filter and design it with MATLAB's `sptool` . Select an appropriate cut-off frequency. Extract the filter coefficients by using `sprintf('%2.10f,\n',filtercoefficients)` . Use the necessary filter functions (provided in the project) and extend the source code in file `main.cpp` . Build and run the program. Does it reduce/eliminate the distortion for the sine-wave signal? Again vary the frequency and describe your observations.

5. Now also put a low-pass filter at the input of the decimator before sampling rate conversion (Filter structure has to be defined twice in the code.). Does this reduce/eliminate the distortion for broadband signals? Why/why not? Is the low-pass filter at the output still necessary? Describe your observations.

6. Replace the low-pass filters with bandpass filters (e.g., for a decimation/interpolation factor of 4, use a passband from $\frac{f_s}{8}$ to $\frac{f_s}{4}$). What do you observe? Provide spectra in your protocol of a fictitious signal with ramp-shaped spectrum for the different stages of the implemented multirate chain with the bandpass filters, i.e. a spectrum at the input, after the first bandpass filter, after down-sampling, after up-sampling, and after the bandpass filter at the output (similar as in Figure 1 and Figure 2).

2 Multirate system with 2-channels

Based on the insights from above, we build a two channel multirate system depicted in Figure 3. The basic structure of the system consists of a low-pass and a high-pass branch. The filters are constructed in a way that the frequency band $\theta \in [0, \pi]$ which corresponds to $f \in [0, \frac{f_s}{2}]$ is split up in two equal parts mirrored at $\theta = \frac{\pi}{2}$ ($f = \frac{f_s}{4}$). The frequency responses of the low-pass filter $H_L(e^{j\theta})$ and the high-pass filter $H_H(e^{j\theta})$ satisfy the following symmetry equation

$$H_H(e^{j\theta}) = H_L(e^{j(\pi+\theta)}).$$

For the corresponding transfer functions we get

$$H_H(z) = H_L(-z)$$

In the time domain, this leads to the following equation for the impulse responses

$$h_H[n] = (-1)^n h_L[n].$$

From this equation we can see that the impulse responses of the two analysis filters differ only in the sign of every other value (for n is odd). Like for the analysis filters, we demand for the synthesis filters that

$$G_H(z) = G_L(-z),$$

and for the impulse responses

$$g_H[n] = (-1)^n g_L[n]$$

In each stage of the synthesis filter bank the sampling rate of the signals has to be increased again.

To ensure that all aliasing components are canceled out we have to decimate the high-pass and the low-pass channel alternately (`OFFSET=1` ; shift by one sample) and choose the same coefficients for the synthesis filter as for the analysis filter: $G_L(z) = H_L(z)$, $G_H(z) = H_H(z)$. Another possibility is to decimate synchronously (`OFFSET=0`), but then we have to invert the high-pass channel $G_H(z) = -G_L(-z)$ to avoid aliasing. More details about this are provided in one of the signal processing lectures.

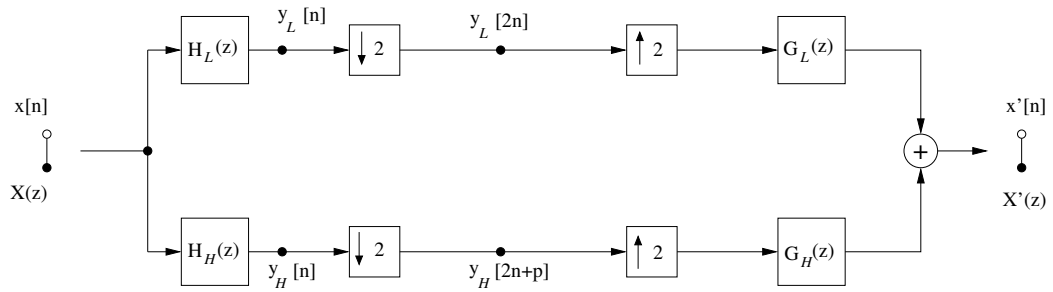

Figure 3: Multirate system with 2 channels.

2.1 Experiment 2: Construction of multirate system

Equipment: PC, raspberry pi, signal generator, oscilloscope
Software: NetBeans, MATLAB, download and unzip `Unit6.zip` from www.spsc.tugraz.at

1. In this experiment, we build the multirate system with two channels shown in Figure 3.

2. Design a halfband filter (low-pass part) with MATLABs sptool (recommended parameters: Kaiser window FIR, order = 18, $\beta = 5$), choose an appropriate cut-off frequency.

3. Use this filter in the provided MATLAB script `test_qmf.m` . This script is an implementation of the system in Figure 3. Analyze this script. Put at the end of the script the commands `figure` and `plot(y_)` . What do you observe? How is alias free reconstruction ensured (alternating decimation/interpolation vs. non-alternating decimation/interpolation but negated high-pass channel). Build a system where aliasing occurs and examine the output signal for a sweep.

4. This MATLAB script has to be implemented on the raspberry pi. Please use the filter coefficients as used in MATLAB.

5. Connect the signal generator, the raspberry pi, and the oscilloscope in a way to be able to measure frequency responses using sweeps.

6. In a first step, implement the low-pass and the high-pass filter at the input – the analysis stage. Determine the frequency response of the low-pass and the high-pass filter. Is the frequency response an exactly mirrored version of the one of the low-pass? If not, why?

7. Extend the program to realize a 2 channel analysis/synthesis filterbank using these filters and the decimators/interpolators. Add the missing code and include the necessary files to the project (they are provided in the same folder). Use the provided `block.decimate()` and `block.interpolate()` functions to implement the decimation/interpolation. Consider `DECFACT` and `INTFACT` .

8. Determine the frequency response of the overall system. Investigate the three cases: the above mentioned cases where aliasing is avoided and one case where aliasing occurs. Provide the figures and explanations in your report.

References

[1] Crochiere, Ronald E. and Rabiner, Lawrence R., "Multirate Digital Signal Processing", Prentice-Hall, Inc., 1983.

[2] Vary, P., Heute, U. and Hess, W., "Digitale Sprachsignalverarbeitung", B.G. Teubner Stuttgart, 1998

[3] Arrowood, J., Randolph, T., Smith, M.J.T., "Filter Bank Design", In: The Digital Signal Processing Handbook, Pages: 36-1–36-17, Editors: Madisetti, V.K., Williams, D.B., CRC Press, 1998

[4] Peevers, Alan W., "A Real Time 3D Signal Analysis/Synthesis Tool Based on the Short Time Fourier Transform", MS Thesis, University of California, Berkeley, `http://www.cnmat.berkeley.edu/~alan/MS-html/MSv2_ToC.html`